Theoretical Computer Science 3 (1977) 105–117.

COMPLETE PROBLEMS FOR DETERMINISTIC POLYNOMIAL TIME

Neil D. JONES and William T. LAASER

Computer Science Department, University of Kansas, Lawrence, KS 66045, U.S.A.

Communicated by Ronald Book
Received March 1975[2]
Revised March 1976

Log space reducibility allows a meaningful study of complexity and completeness for the class $\mathcal{P}$ of problems solvable in polynomial time (as a function of problem size). If any one complete problem for $\mathcal{P}$ is recognizable in $\log^k(n)$ space (for a fixed k), or requires at least n^c space (where c depends upon the program), then all complete problems in $\mathcal{P}$ have the same property. A variety of familiar problems are shown complete for $\mathcal{P}$, including context-free emptiness, infiniteness and membership, establishing inconsistency of propositional formulas by unit resolution, deciding whether a player in a two-person game has a winning strategy, and determining whether an element is generated from a set by a binary operation.

1. Introduction

The results of Cook and Karp [3, 9] aroused considerable interest for at least two reasons. First, the answer to a long-standing open question which had seemed peculiar to automata theory—whether deterministic and nondeterministic polynomial-time-bounded Turing machines are equivalent in power—was seen to be exactly equivalent to determining whether any of a wide variety of familiar combinatorial problems can be solved by polynomial-time algorithms. Second, the existence of *complete* problems for $\mathcal{NP}$[1] made it possible to replace a question about an entire class of problems by a question about a single representative. Thus all of these combinatorial and automata-theoretic problems are essentially restatements of a single problem, such as: "can satisfiability of a propositional formula be decided in polynomial time?"

Another long-standing open question is the relation between time and space-bounded computation. In particular a Turing machine operating in time $T(\cdot)$ also has space $T(\cdot)$ available for computation; but it is not known whether such a machine may in general be replaced by one which operates in the much smaller space bound of $\log T(\cdot)$.

The main purpose of this paper is to show that several familiar problems are complete for $\mathcal{P}$, the class of languages recognizable in deterministic polynomial

[1] $\mathcal{NP}$ denotes the class of all languages which can be recognized in polynomial time by nondeterministic Turing machines.

[2] A preliminary version of this paper appeared in *Proc. Sixth ACM Symp. on Theory of Computing* (1974) 40–46.

time. Any such language has the property that if it is recognizable in space $\log^k(\cdot)$, then *every* language in $\mathcal{P}$ is so recognizable. Thus a problem complete for $\mathcal{P}$ will serve to differentiate those sets in $\mathcal{P}$ which are not recognizable in logarithmic space from those which are, provided such differentiation is possible. A problem with these properties was first presented by Cook in [4], concerning solvable path systems.

The reducibility used herein is the log space reducibility of [8], which is a many-one reducibility as used by Karp, rather than the Turing reducibility of Cook. The first author, and Stockmeyer and Meyer independently observed that the reductions in [9] and [3] were all of this type, which is a refinement of polynomial time reducibility. Clearly such a refinement is necessary in order to study completeness in $\mathcal{P}$, for all problems in $\mathcal{P}$ are equivalent with respect to unrestricted polynomial time reducibility.

Full proofs that the various constructions below can be carried out in log space will not be given; the details are tedious but straightforward. An appropiate set of tools for doing such constructions is provided in the elegant characterization of log-space computable functions provided by Lind and Meyer [10]. In fact it appears that the constructions given herein are also expressible by log-rudimentary functions, an even more restricted class defined in [8].

2. Turing machine and reducibility definitions

By a Turing machine we mean a device with a finite-state control, a two-way read-only input tape with endmarkers, and a single two-way read-write work tape, which halts whenever it enters a final, or accepting, state. The machine is assumed to be deterministic unless otherwise specified; detailed definitions may be found in [7].

Throughout this paper Σ will denote an alphabet with at least two symbols, and $S(\cdot)$ and $T(\cdot)$ will denote nonzero functions on the natural numbers not smaller than the function $\log(\cdot)$. Let Z be a Turing machine with a read-only two-way input tape, and a separate read/write work tape. By definition *Z operates in space $S(\cdot)$* iff, whenever it is started with any $x \in \Sigma^*$ on its input tape, it will never scan more than $S(|x|)$ symbols on its work tape (where $|x|$ denotes the number of symbols in x). In addition we require that Z halts for all inputs. The class DSPACE(S) is defined to be the collection of all languages which are recognizable by deterministic Turing machines which operate in space $S(\cdot)$; NSPACE(S) is defined analogously for nondeterministic Turing machines.

Similarly, a one-tape Turing machine *Z operates in time $T(\cdot)$* iff, given any input $x \in \Sigma^*$, it will make no more than $T(|x|)$ transitions. DTIME(T) and NTIME(T) are the classes of languages recognized by deterministic and nondeterministic Turing machines, respectively, which operate in time $T(\cdot)$.

Language classes of particular interest are the following:

$$\mathscr{L} = \mathrm{DSPACE}(\log(\cdot)), \qquad \mathscr{L}^k = \mathrm{DSPACE}(\log^k(\cdot)),$$

$$\mathscr{P} = \bigcup_{i=1}^{\infty} \mathrm{DTIME}(n^i), \qquad \mathscr{NP} = \bigcup_{i=1}^{\infty} \mathrm{NTIME}(n^i).$$

(By $\log^k(\cdot)$ we mean the function $\lambda n(\log n)^k$; note that the classes $\mathscr{P}$ and $\mathscr{NP}$ remain the same if defined by Turing machines with only a single tape.)

Let $f: \Sigma^* \to \Sigma^*$ be a function. We define $f(\cdot)$ to be *computable in space* $S(\cdot)$ if and only if there is a Turing maching, Z, as in the definition of DSPACE(S) but additionally equipped with a write-only one-way output tape, such that if Z is given any input $x \in \Sigma^*$, Z will eventually halt with $f(x)$ on its output tape, having scanned no more than $S(|x|)$ work tape symbols.

Let $L, L' \subseteq \Sigma^*$. L is (log space) *reducible* to L' (written $L \leqslant L'$) if and only if there is a $\log(\cdot)$-space computable function $f: \Sigma^* \to \Sigma^*$ such that $\forall x \in \Sigma^*$ $(x \in L \Leftrightarrow f(x) \in L')$. A language $L \subseteq \Sigma^*$ is *complete* for a class of languages $\mathscr{C}$ over Σ iff $L \in \mathscr{C}$ and $L' \leqslant L$ for any language L' in $\mathscr{C}$. $\mathscr{C}$ is *closed under (log space) reducibility* iff $L \in \mathscr{C}$ and $L' \leqslant L$ implies $L' \in \mathscr{C}$, for any languages $L, L' \subseteq \Sigma^*$.

Lemma 1. *The reducibility relation* $\leqslant$ *is reflexive and transitive.*

Lemma 2. *$\mathscr{L}$, $\mathscr{L}^k$, $\mathscr{P}$ and $\mathscr{NP}$ are closed under reducibility.*

Lemma 3. *If $L \in \mathscr{L}^k$ is complete for $\mathscr{P}$, then $\mathscr{P} \subseteq \mathscr{L}^k$. (This result is implicit in* [4].)

Proofs. Lemma 1 is Theorem 2 of [8]. The closure of $\mathscr{L}$ and $\mathscr{L}^k$ under reducibility follows from Lemma 3 of [8]; the proof method of that lemma also yields the closure of $\mathscr{P}$ and $\mathscr{NP}$ under reducibility. Now let $L' \in \mathscr{P}$ be arbitrary, and let $L \in \mathscr{L}^k$ be complete for $\mathscr{P}$. Then $L' \leqslant L$ by definition, and so $L' \in \mathscr{L}^k$ by Lemma 2. □

Lemma 3 is of interest chiefly because of the well-known conjecture that $\mathscr{P} \subseteq \mathscr{L}^2$. It appears to these writers that the following conjecture is at least as likely.

Conjecture 4. *If L is any complete set for $\mathscr{P}$, there is a constant $c > 0$ such that L is not in* DSPACE(n^c).

Clearly if one set complete for $\mathscr{P}$ has this property, then *all* complete sets will (although each will have its characteristic value of c).

Lemma 5. *If $L \subseteq \Sigma^*$ is complete for $\mathscr{P}$, then $\Sigma^* - L$ is also complete for $\mathscr{P}$.*

Proof. Assume $L \subseteq \Sigma^*$ is complete for $\mathscr{P}$. Then $\Sigma^* - L$ is clearly in $\mathscr{P}$. Now for arbitrary L', $L' \in \mathscr{P}$ implies $\Sigma^* - L' \in \mathscr{P}$ and so $\Sigma^* - L' \leq L$ by completeness of L. But $\Sigma^* - L' \leq L$ immediately implies $L' \leq \Sigma^* - L$. □

3. The problems

We now list the problems to be shown complete for $\mathscr{P}$, and give necessary definitions of terminology.

(1) UNIT [1]

Given: A propositional formula $\mathscr{F}$ in conjunctive normal form.

To determine: Whether the empty clause □ (indicating a contradiction) may be deduced from $\mathscr{F}$ by unit resolution. [Definition. If clauses F and G have the forms $F = A \vee B_1 \vee \ldots \vee B_m$ and $G = \sim A \vee C_1 \vee \ldots \vee C_n$, the *resolvent* of F and G is $B_1 \vee \ldots \vee B_m \vee C_1 \vee \ldots \vee C_n$. This is called the *unit resolvent* in case $m = 0$ or $n = 0$. A *deduction by (unit) resolution* from $\mathscr{F}$ is a sequence $F_1, F_2, \ldots, F_p$ in which each F_i is either a clause from $\mathscr{F}$, or follows from two earlier clauses by (unit) resolution. (A clause is considered as an unordered set of variables or their complements; resolution is commutative by definition.)]

(2) GEN

Given: A set X, a binary operation $\cdot$ on X, a subset $S \subseteq X$, and an element $x \in X$.

To determine: Whether x is contained in the smallest subset of X which contains S and is closed under $\cdot$.

(3) PATH (Cook [2])

Given: A path system $\mathscr{S} = (X, R, S, T)$ $(S \subseteq X, T \subseteq X, R \subseteq X \times X \times X)$.

To determine: Whether there is an admissible node in S. [Definition. x is admissible iff $x \in T$ or $\exists y, z \in X$ such that $(x, y, z) \in R$ and both y and z are admissible.]

(4) CFØ [7]

Given: A context-free grammar $G = (N, T, P, S)$.

To determine: Whether $L(G) = \emptyset$.

(5) CF∞

Given: A context-free grammar $G = (N, T, P, S)$.

To determine: Whether $L(G)$ is infinite.

(6) CFMEMBER

Given: A context-free grammar $G = (N, T, P, S)$ and $x \in T^*$.

To determine: Whether $S \stackrel{*}{\Rightarrow} x$.

(7) GAME

Given: A two-player game $\mathcal{G} = (P_1, P_2, W_0, s, M)$.

To determine: Whether s is a winning position for the first player. [Definition. A *two-player game* is a 5-tuple (P_1, P_2, W_0, s, M) with $P_1 \cap P_2 = \emptyset$, $W_0 \subseteq P_1 \cup P_2$, $s \in P_1$, and $M \subseteq P_1 \times P_2 \cup P_2 \times P_1$. Informally, P_1 (or P_2) is the set of positions in which it is player one's (or player two's) turn to move. W_0 denotes the set of positions in which player one has an immediate win, and s is the starting position. M defines the set of allowable moves; if $(p, q) \in M$ and $p \in P_1$ (or P_2) then player one (or two) may move from position p to position q in a single step. A position x is *winning* for player one iff x is in W_0, or $x \in P_1$ and $(x, y) \in M$ for some winning position y, or $x \in P_2$ and y is winning for every move (x, y) in M.]

4. Completeness of the problems

If $P(\cdot)$ is a predicate and $C(x)$ is a formula containing x, the notation $\bigvee_{P(x)} C(x)$ serves to abbreviate to the formula $C(x_1) \vee C(x_2) \vee \ldots \vee C(x_n)$, where $\{x_1, x_2, \ldots, x_n\} = \{x \mid P(x) \text{ is true}\}$. $\bigwedge_{P(x)} C(x)$ is used similarly for $C(x_1) \wedge \ldots \wedge C(x_n)$.

Lemma 6. *UNIT is in* $\mathcal{P}$.

Proof. To show that UNIT $\in \mathcal{P}$, we consider the following algorithm to test membership in UNIT. Given a propositional formula $\mathcal{F}$ in conjunctive normal form

(1) Let S = an ordered set consisting of the unit clauses in $\mathcal{F}$, i.e., those consisting of a single literal. Let T = the set of non-unit clauses in $\mathcal{F}$ (we may assume that S and T have some implicit ordering such as the order they appear in $\mathcal{F}$).

(2) **while** $S \neq \emptyset$ **do**

u = the first literal in S;
if u cannot be unit resolved against any clause in $S \cup T$
then $S = S - \{u\}$
else w = the resolvent of u with the first clause v in $S \cup T$
with which u may be unit resolved;
delete v from $S \cup T$;
if $w = \square$ **then** answer "$\mathcal{F}$ yields $\square$ by unit resolution" and halt;
if w is a unit clause **then** add it to the end of S
else add it to the end of T

(3) Answer "$\mathcal{F}$ does not yield $\square$ by unit resolution".

Note that in the **else** part of step 2 of the algorithm, we may delete v since w is more restrictive than v (by the "subsumption principle" [1]).

Now the loop above is executed at most k^2 times, where k is the number of clauses in $\mathscr{F}$. Therefore, the algorithm works in polynomial time.

If $\mathscr{F} \in$ UNIT, then the above algorithm will yield $\square$ since the order of resolution is unimportant, and all possible unit resolutions are performed. Furthermore, since resolution is sound, if the algorithm yields $\square$, then $\mathscr{F}$ is unsatisfiable. Therefore $\mathscr{F} \in$ UNIT iff the algorithm answers yes. Thus UNIT $\in \mathscr{P}$. $\square$

Now suppose Z is a deterministic one-tape Turing machine which operates in at most time n^k for inputs of length n. We define a propositional formula $\mathscr{F}(x)$ corresponding to input x, similar in structure and purpose to $A(w)$ in [3].

Without loss of generality Z has initial state p, accepting state q and rejecting state r such that Z eventually reaches state q or r and remains in that state without terminating, scanning a blank B at its starting position; and Z never moves to the left of its starting position.

Let Σ, Γ and K denote the input, tape and state alphabets, respectively; we represent an instantaneous description (ID for short) by a string α in $\Gamma^*(K \times \Gamma)\Gamma^*$. Let $x = a_1 a_2 \ldots a_n$ where each $a_i \in \Sigma$, and let $w = n^k$ (the maximum number of steps allowed Z). Then by definition

$$\mathscr{F}(x) = \mathscr{J}_0 \wedge \mathscr{J}_1 \wedge \ldots \wedge \mathscr{J}_w$$

where $\mathscr{J}_0, \ldots, \mathscr{J}_w$ are defined below. For each $a \in \Gamma \cup (K \times \Gamma)$, i and t $(0 \leqslant i \leqslant n^k + 1, 0 \leqslant t \leqslant n^k)$ the propositional variable $P^a_{i,t}$ expresses the statement "a is the ith symbol of α_t", where $\alpha_0 \vdash \alpha_1 \vdash \ldots \vdash \alpha_w$ is the computation by Z when given input x.

In the following formulas a, b and c range over all of $\Gamma \cup (K \times \Gamma)$, and i and t range from 1 to w. $\mathscr{J}_0$ and $\mathscr{J}_w$ are defined by:

$$\mathscr{J}_0 = P^{(p,a_1)}_{1,1} \wedge P^{a_2}_{2,1} \wedge \ldots \wedge P^{a_n}_{n,1} \wedge P^{B}_{n+1,1} \wedge \ldots \wedge P^{B}_{w,1} \wedge \bigwedge_t (P^{B}_{0,t} \wedge P^{B}_{w+1,t}),$$

$$\mathscr{J}_w = \sim P^{(q,B)}_{1,w}.$$

The first part of $\mathscr{J}_0$ describes the initial configuration, and the second part specifies that positions 0 and $w+1$ of all IDs are blank. $\mathscr{J}_w$ in effect states that Z rejects input x.

Let $f : (\Gamma \cup (K \times \Gamma))^3 \to \Gamma \cup (K \times \Gamma)$ be the finite function such that if positions $i-1$, i and $i+1$ of α_t $(1 \leqslant i < w)$ contain a, b and c respectively, then position i of α_{t+1} must contain $f(a, b, c)$. For example, if Z in state p_1 reading a "b" prints a "d", goes to state p_2, and moves to the right, then for all $a, c \in \Gamma$,

$$f(a,(p_1,b),c)=d,$$

$$f((p_1,b),a,c)=(p_2,a),$$

$$f(a,c,(p_1,b))=c.$$

If Z in state p_1 reading a "b" prints a "d", goes to state p_2 and moves to the left, then for all $a, c \in \Gamma$,

$$f(a,(p_1,b),c)=d,$$

$$f((p_1,b),a,c)=a,$$

$$f(a,c,(p_1,b))=(p_2,c).$$

Finally, for all $a, b, c \in \Gamma$,

$$f(a,b,c)=b.$$

The determinism of Z ensures that f is single-valued. $\mathcal{I}_t$ expresses the requirement that the ith symbol of the ID α_{t+1} is appropriately determined by the $(i-1)$st, ith and $(i+1)$st symbols of α_t, according to function f. Thus

$$\mathcal{I}_t = \bigwedge_i \bigwedge_{a,b,c} (P^a_{i-1,t} \wedge P^b_{i,t} \wedge P^c_{i+1,t} \rightarrow P^{f(a,b,c)}_{i,t+1})$$

which is clearly equivalent to

$$\mathcal{I}_t = \bigwedge_i \bigwedge_{a,b,c} (\sim P^a_{i-1,t} \vee \sim P^b_{i,t} \vee \sim P^c_{i+1,t} \vee P^{f(a,b,c)}_{i,t+1}).$$

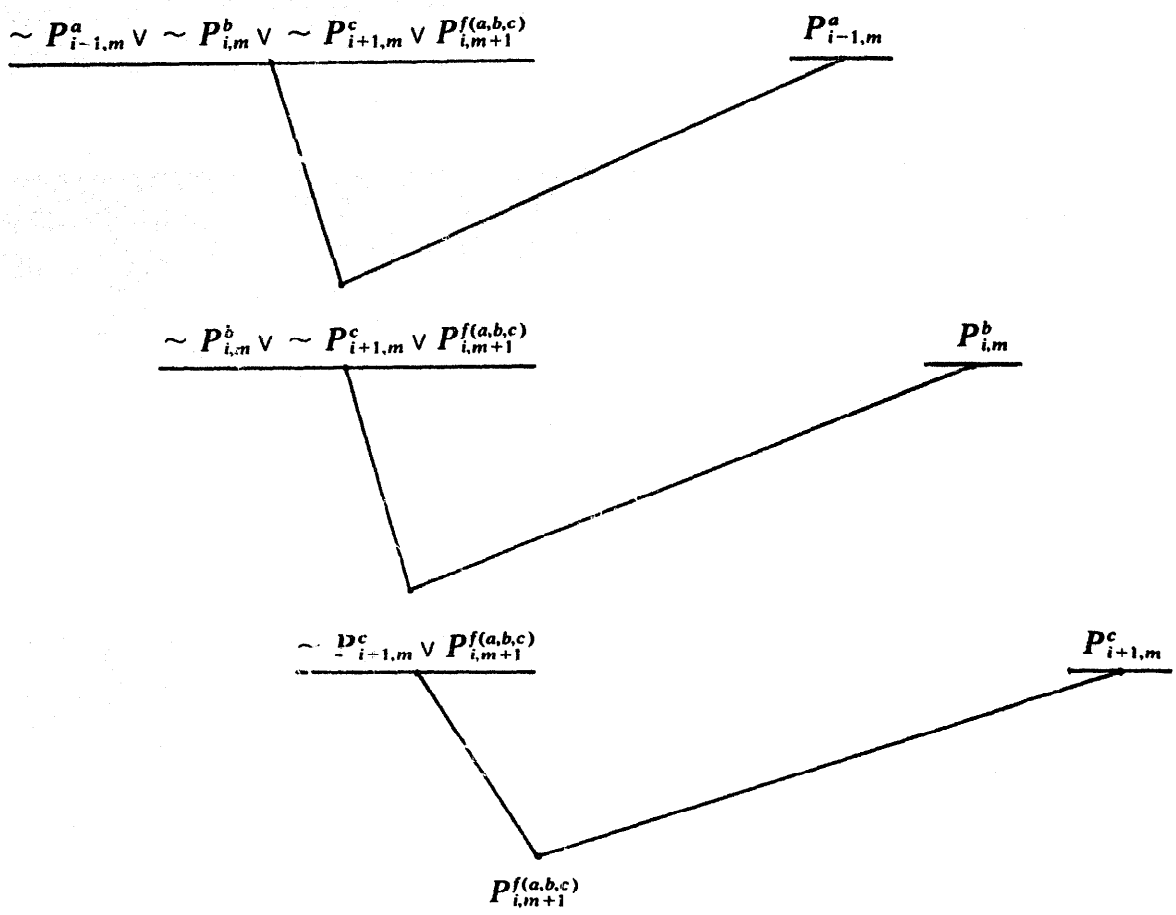
Note that $\mathcal{F}(x)$ is in conjunctive normal form.

Lemma 7. *Let $\alpha_1 \vdash \alpha_2 \vdash \ldots \vdash \alpha_w$ be the computation by Z on $x \in \Sigma^*$. For each $a \in \Gamma \cup (K \times \Gamma)$, position i and time t, $\{1 \leq i \leq w, 1 \leq t \leq w\}$, the unit clause $P^a_{i,t}$ is deducible from $\mathcal{I}_0 \wedge \ldots \wedge \mathcal{I}_{t-1}$ by unit resolution iff a is the ith symbol of α_t.*

Proof. The proof is by induction on t. The claim is trivially true for $t = 1$. Assume the claim is true for $t = m$.

Given an i $(0 < i < w+1)$, if a, b, c in $\Gamma \cup (K \times \Gamma)$ are the symbols at position $i-1$, i, $i+1$, respectively in α_m, then the unit clauses $P^a_{i-1,m}$, $P^b_{i,m}$, and $P^c_{i+1,m}$ are all deducible from $\mathcal{I}_0 \wedge \ldots \wedge \mathcal{I}_{m-1}$ by the induction hypothesis. But $\sim P^a_{i-1,m} \vee \sim P^b_{i,m} \vee \sim P^c_{i+1,m} \vee P^{f(a,b,c)}_{i,m+1}$ is a clause in $\mathcal{I}_m$. Therefore, $P^{f(a,b,c)}_{i,m+1}$ may be deduced by the following diagram.

Furthermore, it is easy to see that no positive single literal clause may be deduced from $\mathcal{I}_0 \wedge \ldots \wedge \mathcal{I}_m$ unless it is deduced in a manner similar to that above. But by the definition of f, $f(a, b, c)$ is exactly the contents of position i in α_{m+1}. Therefore $P^a_{i,m+1}$ is deducible from $\mathcal{I}_0 \wedge \ldots \wedge \mathcal{I}_m$ by unit resolution iff a is the symbol in the ith position of α_{m+1}. □



Theorem 8. *UNIT is complete for* $\mathcal{P}$.

Proof. By Lemma 6, UNIT is in $\mathcal{P}$, so it suffices to show that $L \leqslant \text{UNIT}$ for an arbitrary L in $\mathcal{P}$. Since L is in $\mathcal{P}$, there is a one-tape Turing machine Z and a positive integer k such that Z accepts L in time n^k.

Now let $x \in \Sigma^*$ be arbitrary, let $n = |x|$, and let $\alpha_1 \vdash \ldots \vdash \alpha_{n^k}$ be the computation by Z on x as input. Then $x \in L$ iff (q, B) is the first position of α_{n^k}, so $x \in L$ iff $P^{(q,B)}_{1,n^k}$ is deducible. But $\sim P^{(q,B)}_{1,n^k}$ is a clause in $\mathcal{I}_{n^k}$. Thus the $\square$ is deducible from $\mathcal{F}(x)$ by unit resolution. Therefore, $x \in L$ implies $\mathcal{F}(x) \in \text{UNIT}$.

Conversely, if $\square$ can be deduced, then so can $P^{(q,B)}_{1,n^k}$. But then by the claim, Z on input x at time n^k must be in state q. Thus $x \in L$. Therefore $\mathcal{F}(x) \in \text{UNIT}$ implies $x \in L$.

It is straightforward to show that $\mathcal{F}(x)$ is calculable from x in $\log(\cdot)$ space, so $L \leqslant \text{UNIT}$. Thus UNIT is complete for $\mathcal{P}$. $\square$

Remark. When applied to nondeterministic Turing machines, this construction naturally leads to a $\mathcal{I}(x)$ containing clauses of the form $F \wedge G \wedge H \rightarrow I \vee J$. As in Cook [3], this $\mathcal{I}(x)$ will be satisfiable iff Z accepts x. Since resolution is a complete inference rule for propositional logic, we may state the following:

A language L is in $\mathcal{NP}$ iff there is a log-space computable function $\mathcal{F}$ such that any x is in L iff $\square$ may not be deduced from $\mathcal{F}(x)$ by resolution.

By use of Lemma 5, Theorem 8 may be recast into very similar form:

A language L is in $\mathcal{P}$ iff there is a log-space computable function $\mathcal{F}$ such that any x is in L iff $\square$ may not be deduced from $\mathcal{F}(x)$ by *unit* resolution.

From another point of view, nondeterministic computations correspond to formulas of the kind $F \wedge G \wedge H \rightarrow I \vee J$, and deterministic computations correspond to formulas of the kind $F \wedge G \wedge H \rightarrow I$ (which are also known as Horn formulas). Further, it is known that unit resolution is a complete deduction rule for Horn formulas [6].

We now discuss the GEN problem.

Note: in the following it is assumed that $\cdot$ is given explicitly by a multiplication table.

Corollary 9. *GEN is complete for* $\mathcal{P}$.

Proof. GEN is easily seen to be in $\mathcal{P}$, by a simple marking algorithm.

Now let $L \in \mathcal{P}$ be arbitrary, and let $\mathcal{F}(\cdot)$ be as in the proof of the preceding theorem. For an arbitrary $x \in \Sigma^*$, define

$$X = \{\#\} \cup \{\mathcal{J} \mid \mathcal{J} \text{ is a subclause in } \mathcal{F}(x)\},$$

$$T = \{\mathcal{J} \mid \mathcal{J} \text{ is a clause in } \mathcal{F}(x)\},$$

$$\mathcal{J}_1 \cdot \mathcal{J}_2 = \begin{cases} \mathcal{H} & \text{if } \mathcal{H} \text{ is the unit resolvent of } \mathcal{J}_1 \text{ and } \mathcal{J}_2, \text{ or} \\ \# & \text{otherwise,} \end{cases}$$

$$w = \square, \quad \text{the empty clause.}$$

Clearly, $w = \square$ is generated by T iff $\mathcal{F}(x) \in \text{UNIT}$. The fact that no clause in $\mathcal{F}(x)$ has more than 4 literals ensures that X has at most one plus 16 times as many element as $\mathcal{F}$ has clauses. Thus X, $\cdot$, T and w can be obtained in log space. $\square$

Note that in this case $\cdot$ is commutative but not associative. In fact, the GEN problem for associative $\cdot$ may be seen to be complete for the sets recognizable in nondeterministic $\log(\cdot)$ space.

The following result is known from Theorems 1 and 2 of Cook [4], although it is not expressed in terms of completeness. The proof method is rather different.

Theorem 10. *PATH is complete for* $\mathcal{P}$.

Proof. PATH $\in \mathcal{P}$ by Theorem 1 of [4]. Thus to show that PATH is complete for $\mathcal{P}$, it suffices to show that GEN $\leq$ PATH. Let X, $\cdot$, T, and w be given. Construct a path system $S = (X, R, \{w\}, T)$ where R is defined as follows:

$$\forall x, y, z \in X \quad (x, y, z) \in R \text{ iff } x = y \cdot z.$$

It is now a trivial matter to see that $S \in \text{PATH}$ iff w is generated by T. Furthermore, this reduction can easily be done in $\log(\cdot)$ space. □

The following result for CFMEMBER may be surprising in view of Lemma 3 and the fact that any single context-free language may be parsed in $\log^2(\cdot)$ space [12]. The apparent implication that any language in $\mathcal{P}$ can be recognized in $\log^2(\cdot)$ space is invalid, though, because the parsing procedure requires a Chomsky normal form grammar. The difficulty is that conversion to Chomsky form seems to require linear space in order to eliminate ε-productions. From another point of view, the problem is that the CFMEMBER problem requires both a grammar *and* a terminal string as input, while the passing algorithm of [12] assumes a fixed grammar.

Corollary 11. *CF∅ and CFMEMBER are complete for* $\mathcal{P}$.

Proof. CF∅ is in $\mathcal{P}$ by standard methods; and CFMEMBER is in $\mathcal{P}$ by Younger's parsing algorithm [7]. Again, to show completeness it suffices to show GEN $\leqslant$ CF∅ and GEN $\leqslant$ CFMEMBER, which we show simultaneously.

Let X, $\cdot$, T and w be given. Construct the context-free grammar $G = (X, \{a\}, P, w)$, where

$$P = \{x \to yz \mid y \cdot z = x\} \cup \{x \to \varepsilon \mid x \in T\}.$$

It is easy to see that $L(G) = \{\varepsilon\}$ iff w is generated by T. Consequently $L(G) \neq \emptyset$ and $\varepsilon \in L(G)$ iff w is so generated. The reduction can certainly be done in $\log(\cdot)$ space. □

Corollary 12. *CF∞ is complete for* $\mathcal{P}$.

Proof. The natural approach is to first remove useless nonterminals and ε-productions, and apply a straightforward algorithm to the resulting grammar. This method fails, however, due to the fact that removal of ε-productions may increase the grammar's size exponentially.

We first show CF∞ $\in \mathcal{P}$ by use of the following fact. Given a context-free grammar $G = (N, T, P, S)$, $G \in$ CF∞ iff there exist $A \in N$; $x, y \in (N \cup T)^*$; and $t, u, v \in T^*$ such that $uv \neq \varepsilon$ and

(a) $S \Rightarrow^* xAy$,

(b) $A \Rightarrow^* uAv$, and

(c) $A \Rightarrow^* t$.

These conditions may be tested by first removing inaccessible or useless symbols and productions, thus guaranteeing that conditions (a) and (c) are satisfied for any remaining nonterminals B. Now define for each nonterminal A,

$$U_A = \{B \in N \mid \exists \alpha, \beta \in (N \cup T)^* A \Rightarrow^* \alpha B \beta\},$$

$$M_A = \{B \in N \mid \exists \alpha, \beta \in (N \cup T)^* A \Rightarrow^* \alpha B \beta \text{ and } \alpha\beta \neq \varepsilon\}.$$

Clearly (b) is satisfied iff $A \in M_A$. Further, the sets U_A and M_A may be built up in parallel, by scanning them and G repeatedly. This requires only a polynomial amount of time, so CF∞ is in $\mathcal{P}$.

By Lemma 5, since CF∅ is $\mathcal{P}$ complete, so is $\Sigma^* - \text{CF}\emptyset$. Therefore, to show that CF∞ is $\mathcal{P}$ complete, it suffices to show that $(\Sigma^* - \text{CF}\emptyset) \leq \text{CF}\infty$. (*Note*: with no loss of generality we may assume that every string in Σ^* specifies some context-free grammar.) Given a context-free grammar $G = (N, T, P, S)$, define a context-free grammar $G' = (N \cup \{X\}, T \cup \{a\}, P', S)$, where $X \notin N \cup T$ and P' consists of all productions in P with terminals replaced by the nonterminal X, all productions of the form $A \to \varepsilon$ replaced by $A \to X$, and the following two productions added:

$$X \to aX, \qquad X \to a.$$

This reduction may easily be done in log space, and clearly $G \in \Sigma^* - \text{CF}\emptyset$ iff $G' \in \text{CF}\infty$. Therefore, $\Sigma^* - \text{CF}\emptyset \leq \text{CF}\infty$. Thus CF∞ is complete for $\mathcal{P}$. □

Finally, we show that determination of whether the first player has a winning strategy in an explicitly given 2-player game is complete for $\mathcal{P}$.

Theorem 13. *GAME is complete for* $\mathcal{P}$.

Proof. First, GAME $\in \mathcal{P}$, since given $\mathcal{I} = (P_1, P_2, W_0, s, M)$, we may decide if $\mathcal{I} \in$ GAME by the marking algorithm naturally implied by the definition, as follows:

(a) $W_0 = W$,

(b) **for** $i = 1, 2, \ldots$ **until** $W_i = W_{i+1}$ **do**

$$W_{i+1} = W_i \cup \{x \in P_1 \mid \exists y (x, y) \in M \wedge y \in W_i\} \cup \{x \in P_2 \mid \forall y [(x, y) \in M \to y \in W_i]\}.$$

We now show GEN $\leq$ GAME. Let X, $\cdot$, T and w be given. Construct the game $\mathcal{I} = (X, X \times X, T, w, M)$, where the allowable moves are given by

$$M = \{(p, (q, r)) \mid q \cdot r = p\} \cup \{((p, q), p) \mid p, q \in X\} \cup \{((p, q), q) \mid p, q \in X\}.$$

We shall show that $\mathcal{I} \in$ GAME iff w is generated from T. The intuitive idea is that player 1 attempts to prove that a node p (initially w) is generated by T, by exhibiting two elements q, r such that $p = q \cdot r$, which he claims to be also generated by T [formally he moves from position p to position (q, r) where $p = q \cdot r$]. On the other hand player 2 attempts to show player 1 to be incorrect by exhibiting an element of the pair which is not generated by T (i.e. player 2 can move from position (p, q) to either position p or position q). Player 1 wins whenever player 2 picks an element which is already in T (and therefore generated by S).

Now let W be as in the definition, and let V be the smallest set which contains T and is closed under $\cdot$

Claim. *A position in X is winning iff it is in V.*

Proof of Claim. (1) Every position in V is winning, since $W \supseteq W_0 = T$, and $W \cap X$ is closed under $\cdot$ as follows. Let $p, q \in W$; now the only moves from (p, q) are to p and to q [namely $((p, q), p)$ and $((p, q), q)$]. Since p and q are both winning, the position (p, q) is also winning. Now $(p \cdot q, (p, q))$ is an allowable move, so $p \cdot q$ is winning by (b) in the definition of W.

(2) We actually show that any winning position is in $V \cup V \times V$, by induction on i in the sequence $W_0, W_1, \ldots$ above. This is certainly true for $i = 0$, since $W_0 = T \subseteq V$; so suppose it true for $i \geq 0$, and consider an arbitrary position in $W_{i+1} - W_i$. Either this is a node p in X, or a pair (p, q) in $X \times X$. In the first case a move $(p, (q, r))$ must be in M for some $(q, r) \in W_i$ such that $p = q \cdot r$. By induction $(q, r) \in V \times V$, so $q \in V$ and $r \in V$; consequently $p \in V$. In the other case whenever $((p, q), r)$ is a move, r must be in W_i. By definition of M this means both p and q are in W_i, and so in V by induction; thus position (p, q) is again in $V \cup V \times V$. □

Therefore, $\mathcal{S} \in$ Game iff w is winning iff $w \in V$ iff w is generated from T. Furthermore, $\mathcal{S}$ can easily be constructed from X, $\cdot$, T and w in $\log(\cdot)$ space. Thus GEN $\leq$ GAME, so GAME is complete for $\mathcal{P}$. □

Remark. The result above assumes that the game is given in explicit form. A rather different result might follow in case the winning positions and allowable moves were given by means of a computational procedure rather than explicitly. *Note added after proof of the theorem*: Even and Tarjan [5] have recently shown that a HEX-like game on graphs is complete for polynomial space, thus verifying the previous remark.

References

[1] C. Chang and R. Lee, *Symbolic Logic and Mechanical Theorem-Proving* (Academic Press, New York, 1973)

[2] S.A. Cook, Path systems and language recognition, *Proc. Second ACM Symp. on Theory of Computing* (1970) 70–72.

[3] S.A. Cook, The complexity of theorem-proving procedures, *Proc. Third ACM Symp. on Theory of Computing* (1971) 151–158.

[4] S.A. Cook, An observation on time-storage trade-off, *J. Comput. System Sci.* 9 (1974) 308–316.

[5] S. Even and R. Tarjan, A combinatorial problem which is complete in polynomial space, *Proc. Seventh ACM Symp. on Theory of Computing* (1975) 66–71.

[6] L. Henschen and L. Wos, Unit refutations and Horn sets, *J. ACM* 21 (4) (1974) 590–605.

[7] J. Hopcroft and J. Ullman, *Formal Languages and Their Relation to Automata* (Addison-Wesley, Reading, Mass., 1969).

[8] N.D. Jones, Space-bounded reducibility among combinatorial problems, *J. Comput. System Sci.* **11** (1) (1975) 68–85.

[9] R.M. Karp, Reducibility among combinatorial problems, in: *Complexity of Computer Computations*, R.E. Miller and J.W. Thatcher, eds. (Plenum Press, N.Y., 1972) 85–103.

[10] J. Lind and A.R. Meyer, A characterization of log-space computable functions, *SIGACT News*, **5** (3) (1973) 26–28.

[11] L.J. Stockmeyer and A.R. Meyer, Word problems requiring exponential time: preliminary report, *Proc. Fifth ACM Symp. on Theory of Computing* (1973) 1–9.

[12] D.H. Younger, Recognition and parsing of context-free languages in time n^3, *Inform. and Control* **10** (2) (1967) 189–208.